AMENDMENTS TO THE SPECIFICATION

Please amend paragraph 0012 as follows:

[0012] The embedded system of a network device 130 may <u>be</u> built from modules such as management clients 120, configuration manager 140, modules 160, and system control module 170, these modules may be designed to be reusable, to minimize costs of software development effort. In some embodiments, a collection of these modules may be reused as whole or in a part to design a new (perhaps different) specialized computing device. The complexity of this task can depend also on a module framework design. A framework may facilitate construction of the target system from the building blocks and give high degree of software reusability from device to device. In particular, embedded system modules may facilitate: (1) orderly initialization, (2) orderly shutdown, and/or (3) orderly command script generation. Orderly Initialization can comprise coordination of modules' activities because some modules always depend on other. Also, as a part of initialization, modules may be configured using permanently stored parameters (a.k.a. registry) and default values. This process can also be coordinated between modules.

Please amend paragraph 0017 as follows:

[0017] Figure 3 shows an example of events during system initialization. After [[an]] loading an embedded system into a computing device, the system control module may be initialized. All other system modules may register 310 with system control module, giving their requirements on initialization, by specifying initialization steps and relationships to other modules. At this time, the modules may also register commands

with CLI 360 and 350. CLI may assign a phase identification (Id) to each command. The initialization steps may include e.g., a resource allocation step, an initialization of used libraries, and an initialization of management interface (CLI, SNMP, etc.). Further, the relationships can define dependencies to other system modules, e.g., IP stack has to be initialized before OSPF router software. Based on registration information, the system control module may determine an inter-module dependency tree 320. The system control module may use the inter-module dependency tree 320 to complete initialization 330, 350, and 360, perform device shutdown 330, 350, and 360, and ensure a proper sequence of generated configuration script 330 and 340. As the last step, each module configuration may be restored as a result of the following actions: (1) batch CLI session is started 330; (2) a sequence of phase Ids may be taken for each module 330; (3) from the command script 340, the commands matching given phase Id can be executed 350; (4) configuration manager can pass new parameters value to the current configuration database 360, and (5) the new parameters are passed to owning modules 360.

Please amend paragraph 0035 as follows:

[0035] The machine-readable medium 800 may comprise instructions for determining an inter-module dependency tree 810 and modifying a module function in accordance with the inter-module dependency tree 830. Determining an inter-module dependency tree 810 can comprise instructions for associating a module command with an inter-module dependency 815. Associating a module command with an inter-module dependency 815 may comprise instructions to build one or more trees comprising each module command associated with a node. In some embodiments, [[One]] <u>one</u> or more inter-module dependency trees may be created and may not be associated with each

other. Many embodiments comprise a module dependency tree associated with each module registered with a system controller. In some embodiments, the lowest member or members of each tree are associated with a phase for initialization and reconfiguration and the highest member or members are associated with a phase for shut down.